08

Manejando el commit y el rollback

Transcripción

[00:00] Hola. En la clase anterior vimos qué podemos hacer para garantizar que la operación sea realizada por completo o no haga nada si ocurre un error. Aprendimos con eso a tomar el control de la transacción con el comando setAutoCommit de la conexión.

[00:14] Pero cuando realizamos esa configuración, llegamos al punto de que la aplicación ya no registra ningún producto más y la idea para nuestra transacción es garantizar que o registramos la cantidad total de producto o no registramos nada.

[00:30] ¿Por qué seguiremos con esta regla? Imaginémonos acá que tenemos dos cuentas bancarias y una de ellas que es la cuenta de origen, quiere hacer una transferencia para la cuenta de destino. En una transferencia entre cuentas, el dinero sale de la cuenta de origen, se sustrae de su saldo y va para la cuenta de destino, que recibe este dinero y lo suma a su saldo.

[00:54] Si en el momento acá de esta transferencia hay un error en la operación, o sea tenemos acá una X. Y cuando la cuenta de origen se bajó su saldo, se hizo la transferencia en el momento de llegar a la cuenta de destino, un error. ¿Ahí qué pasa? ¿La cuenta de origen pierde el dinero? En realidad no. Esa transacción no va a ser concluida pero sí revertida.

[01:26] Como hubo un error acá, al momento de llegar el dinero a la cuenta de destino, el valor que salió de la cuenta de origen debe volver a esta cuenta para que su saldo quede ahí mantenido sin ninguna modificación. En la caja va a s

presentado un error para el usuario informando que hubo un fallo en la transferencia.

[01:44] Ahora imagina si esta operación es ejecutada en dos transacciones distintas. En la primera el dinero sale de la cuenta de origen y se concluye. O sea, chau saldo, ya se fue el dinero que quieras transferir. Y en la segunda, el dinero transferido para la cuenta de destino tiene un error, o sea se pierde el dinero.

[02:05] Ni la cuenta de origen va a tener el dinero de vuelta y ni la cuenta de destino va a tener su dinero que fue transferido de la cuenta de origen. Eso no puede pasar. Por eso nosotros tenemos que garantizar que toda la operación quede dentro de una sola transacción. La cantidad total de productos debe ser registrada por completo por el mismo motivo que una transferencia de dinero entre cuentas debe ser completa por entero o nada cambia.

[02:32] Porque si hay un error en el medio del proceso, todas las operaciones de la base de datos deben ser revertidas. Ahora, vamos para la solución del problema en el código. Cuando trabajamos con el control manual de una transacción, o sea con el AutoCommit(false); nosotros tenemos que agregar explícitamente el comando de commit en el código.

[02:55] El comando con.commit lo vamos a agregar afuera del bloque de do while. Y va a ser acá con.commit(); para garantizar que todos los comandos del loop hayan sido ejecutados correctamente. Ahora sí, estoy levantando la aplicación y cuando vaya a guardar las 60 zapatillas ellas tienen que ser todas guardadas ahí sin ningún problema, sin ocurrir ningún fallo.

[03:21] Vamos a ver acá las zapatillas de básquet, 60 unidades, guardar y ahí tenemos registrado con éxito y acá aparecieron las 60 zapatillas. Un registro con 50 y un registro con 10. Igualmente para garantizar que no tengamos errores, dejar el código con la mejor calidad y cobertura de escenarios de

error, nosotros podemos agregar acá el comando de rollback, que la transacción si ocurrió un fallo.

[03:50] Vamos a hacer lo siguiente acá. Nosotros vamos a agregar un bloque de try catch acá, que va a mantener toda esta lógica de do while y del commit y si hay un catch acá con SQL, una exception en realidad, si hay una exception, cualquiera que sea, nosotros vamos a hacer un con.rollback.

[04:18] Es así. Y al final cerramos la conexión y ya está. O sea, si la ejecución acá tiene un error, él ejecuta registro o cualquier cosa que hay acá dentro del try tiene un error, vamos a caer en el catch, nosotros vamos a hacer un rollback de la transacción, vamos a cerrar la conexión y no hay ningún problema. Nosotros cancelamos la ejecución de estas transacciones.

[04:42] Aprovechando acá yo cerré la conexión pero podemos cerrar también el preparedStatement. Entonces voy a hacer un statement.close() también para tener un mejor control de la transacción. Ahora sí estamos controlando de verdad la transacción del método de registrar un producto.

[05:00] Si sale todo bien, tenemos un commit y si ocurre un error, tenemos un rollback de la transacción. ¿Vamos a probar el rollback? Vamos a agregar aquí, en el ejecutaRegistro una vez más la lógica de if (cantidad < 50) lanzamos un throw new RuntimeException ("Ocurrió un error") Ahí está. Vamos a probarlo una vez más.

[05:33] Voy a levantar la aplicación. Ahora acá yo voy a borrar las zapatillas para intentar agregarlas una vez más. Ahí borré una, borré la otra y ahora vamos a agregar las 60 zapatillas de básquet. 60. Guardar. Registrado con éxito. Mira qué bien. O sea, no hubo nada, algo registrado con éxito. Tenemos un fue insertado el producto de id 23 pero nosotros no tuvimos nada. ¿Por qué?

[06:10] O guardamos todo o no guardamos nada. Para que quede más claro acá nosotros vamos a hacer lo siguiente. Voy a bajar esta aplicación y acá, cuando ocurra el rollback o el commit, yo voy a imprimir en la consola. Acá voy a tener

un ("COMMIT") y en el catch voy a poner un ("ROLLBACK"). Así vamos a ejecutar una vez más y vamos ver que quede más claro, porque vimos que hubo un éxito pero no hubo registro y quedó un poco confuso.

[06:40] Así que acá está la aplicación, no tenemos nada y ahora vamos a agregar las 60 zapatillas de básquet. 60. Guardar. Éxito pero, en la consola tenemos el ROLLBACK. O sea, hubo un error en la ejecución, nada fue guardado y se hizo un rollback de la transacción. Ahora garantizamos que la transacción o guarda todo o no guarda nada.

[07:13] Y con eso aprendimos cómo funciona el control de transacciones de aplicaciones del mercado. Generalmente son códigos así que son creados en aplicaciones reales. Vamos cada vez más mejorando la calidad de nuestro código y nuestra aplicación. Nos vemos en la próxima clase para aprender un recurso nuevo que va a dejar nuestro código aún más sencillo.